Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 1834–1851
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Optimal resilient threshold GQ signatures ☆

Cheng-Kang Chu *, Wen-Guey Tzeng

Department of Computer and Information Science, National Chiao Tung University, Hsinchu 30050, Taiwan

Received 24 June 2005; received in revised form 16 July 2006; accepted 8 November 2006

Abstract

GQ signature scheme is widely used in many cryptographic protocols, such as forward-secure signature scheme, identity-based signature scheme, etc. However, there is no threshold version of this important signature scheme in the open literature. We proposed the first threshold GQ signature scheme. The scheme is proved unforgeable and robust against any adaptive adversary by assuming hardness of computing discrete logarithm modulo a safe prime and existence of fully synchronous broadcast channel. Furthermore, with some modifications, our scheme achieves optimal resilience such that the adversary can corrupt up to a half of the players. As an extension of our work, we provided a threshold identity-based signature scheme and a threshold forward-secure signature scheme, which is the threshold version of the most efficient forward-secure signature scheme up to now.

Keywords: GQ signature scheme; Threshold signature scheme

1. Introduction

A threshold cryptographic protocol involves a set of players together, who each possesses a secret share generated by a prior share generating process, to accomplish a cryptographic task. It provides strong security assurance and robustness against a number of malicious attackers under a threshold. For example, in a (t, l)-threshold signature scheme, as long as $t + 1$ players agree, they can jointly produce a signature for a given message even some other players intend to spoil such process. Also, as long as the adversary corrupts less than $t + 1$ players, it cannot forge any valid signature.

A threshold protocol contains *share distribution* phase and *cryptographic function* phase. The share distribution phase is performed by a dealer (honest dealer model) or by players themselves (distributed key generation model) such that each player obtains a secret share. Then all players perform the cryptographic function phase using their own shares. In the previous works, almost all discrete-log based protocols are under distributed key generation model because we can easily generate this type of keys in distributive way via joint verifiable secret sharing scheme [34]. In contrast, the distributed key generation in a factoring based protocol

☆ This is an extended version of the extended abstract in ACNS'03 [32]. Research supported in part by National Science Council grant 94-2213-E-009-110, Taiwan and Taiwan Information Security Center (TWISC).
* Corresponding author. Tel.: +886 918 629332; fax: +886 3 572 1490.
E-mail addresses: ckchu@cis.nctu.edu.tw (C.-K. Chu), tzeng@cis.nctu.edu.tw (W.-G. Tzeng).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2006.11.001

is much more complex. Most factoring based distributed protocols assume an honest dealer to distribute the secret shares. Our scheme is based on the factoring problem, and the secret shares are distributed by an assumed dealer. We focus on the cryptographic function phase, that is, each player holds a share to compute a signature of the given message.

Guillou and Quisquater [25] proposed an identification scheme. Then the GQ signature scheme is obtained by the standard Fiat–Shamir transformation [15]. GQ signature scheme is widely used in many cryptographic protocols, such as forward-secure signature scheme [26], identity-based signature scheme [12], etc. However, there is no threshold version of this important signature scheme in the open literature. We study a threshold signature protocol based on the GQ signature scheme in this paper. By assuming the existence of fully synchronous broadcast channel, our scheme is secure in the adaptive adversary model. The adaptive adversary, in opposition to the static adversary, means that the adversary chooses players it wants to attack in the execution of protocols, where the static adversary can only choose them before the execution of protocols. Furthermore, with some modifications, our scheme achieves optimal resilience such that the adversary can corrupt up to a half of the players. We also extend our work to the identity based and forward-secure signatures paradigm.

1.1. Related work

Threshold schemes can be generally applied by the secure multiparty computation, introduced by [39,24]. However, these solutions based on the protocols that compute a single arithmetic or Boolean gate are inefficient and impractical. The first general notion of *efficient* threshold cryptography was introduced by Desmedt [13]. It started many studies on threshold computation models and concrete threshold schemes based on *specific* cryptosystems. Here we focus on two popular signature schemes, DSS and RSA signatures.

For the DSS scheme, the first solution was proposed by Cerecedo et al. [8] under a non-standard assumption. Gennaro et al. [23] provided another solution with security relying only on the regular DSS signature scheme. Canetti et al. [7] and Frankel et al. [20] improved the security against the *adaptive* adversary. Jarecki and Lysyanskaya [28] furthermore removed the need of reliable erasures. Jarecki [27] summarized these techniques. These works also contain the distributed key generation protocol such that each player holds a key share before signing.

On the other hand, threshold RSA problem is more interesting. Since the share holders do not know the order of the arithmetic group, the polynomial interpolation is much harder than those of discrete-log based threshold cryptosystems. Desmedt and Frankel [14] provided the first heuristic threshold RSA scheme without security analysis. Later, they extended their work with a security proof [16]. De Santis et al. [11] also proposed another provably secure threshold RSA scheme. Both [16,11] tried to avoid the polynomial interpolation. However, these schemes are complicated and need either interaction or large share sizes. Besides, they do not consider the robustness property. The robust threshold RSA schemes were proposed by Gennaro et al. [22] and Frankel et al. [18]. Subsequently, some more efficient and simpler schemes for threshold RSA in the static adversary model were presented [17,35]. These schemes take an extra layer of secret sharing so that more interactions are needed. Shoup [36] provided a much simpler scheme without any interaction in the partial signature generation phase (also see [30] for a more efficient solution in case of passive attacks). Damgård and Koprowski [10] proposed schemes which are as efficient as Shoup's and use general moduli. Moreover, Damgård and Dupont [9] provided a robust threshold RSA scheme with the same efficiency and general moduli. For the adaptively-secure threshold RSA, there exist some solutions [7,20,21,29] as well. These protocols developed many techniques for designing secure threshold protocols.

There are two types of forward-secure signature schemes. The first type is a general construction based on arbitrary signature schemes. It mainly uses the master public key to certify the public key of the current time period. The second type modifies existing concrete signature schemes. Currently, there are four such forward-secure signature schemes [4,2,26,5]. There are threshold versions for the first two schemes [37,1]. In this paper we propose the threshold version for the third one.

2. Preliminaries

Before presenting our scheme we review the GQ signature scheme, introduce the components of a threshold signature scheme, and define the security of a signature scheme and a threshold signature scheme respectively.

2.1. GQ signature scheme

The three functions of GQ signature scheme are described as follows (security parameter: k_1, k_2). Let $x \in_R X$ denote that x is chosen from X randomly.

(1) *Key generation:* let $n = pq$ be a k_1-bit product of two safe primes and e be a $(k_2 + l)$-bit random value. The secret key of a player is (n, e, s), where $s \in_R Z_n^*$ and the corresponding public key is (n, e, v), where $v = 1/s^e \bmod n$. Note that the user does not need to know p and q and thus n can be used by all users.
(2) *Signing*: let H be a publicly defined cryptographic-strong hash function, such as SHA-1. Given a message M, the signer computes the signature (σ, z) as follows:
   - Randomly select a number $r \in Z_n^*$ and compute $y = r^e \bmod n$ and $\sigma = H(y \| M)$.
   - Compute $z = r \cdot s^\sigma \bmod n$.
(3) *Verification*: given the signature (σ, z) and the message M, the verifier checks whether $H(z^e v^\sigma \bmod n \| M) = \sigma$.

The security of GQ signature scheme is based on the assumption that computing eth root modulo a composite is infeasible without knowing the factors of n.

2.2. Threshold signature scheme

A *threshold signature scheme* consists of the following three components:

(1) *Key generation:* there are two categories in generating keys and distributing shares to the participated players:
   - Honest Dealer model: an honest dealer is responsible for choosing keys and distributing shares to the players.
   - Distributed key generation model: all players compute their keys and shares in a distributed way.

   After key generation, the public/secret key pair is defined and each player holds a share of the secret key.
(2) *Distributed signing*: the main signing procedure can be separated into two phases:
   - Partial signature generation: each player produces a partial signature for the given message M with or without interactions.
   - Signature construction: any one who has a number of valid partial signatures over a threshold can compute a valid signature for M.
(3) *Verification*: everyone can verify the validity of a signature for a message given the public key.

The signing players first run the key generation protocol, and get their secret shares. Whenever a message needs to be signed, all players perform the distributed signing protocol to get the signature.

2.3. Security definitions

Consider a signature scheme secure against existential forgery under an adaptive chosen message attack (CMA), we define the security as follows.

Definition 1 (*Security of signature scheme*). A signature scheme is (t_f, ε)-secure if no adversary outputs a valid forgery with probability at least ε within time t_f under an adaptive chosen message attack.

The security of threshold signature scheme includes both unforgeability and robustness as defined below.

Definition 2 (*Unforgeability*). $A(t, l)$-threshold signature scheme is (t_f, ε)-unforgeable in certain adversarial model if, except a negligible probability ε, it no adversary in that model corrupts up to t players can produce a valid signature on a message that was not signed by any uncorrupted player within time t_f.

Another important property of threshold schemes is *robustness*. It ensures that the protocol can output a correct result as long as the adversary controls at most t players.

Definition 3 (*Robustness*). $A(t,l)$-threshold signature scheme is robust in certain adversarial model if even the adversary in that model controls up to t players, the signature scheme is guaranteed to complete successfully.

A threshold signature scheme is called (t_f,ε)-*secure* if the above two properties are satisfied.

Definition 4 (Security of threshold signature). $A(t,l)$-threshold signature scheme is (t_f,ε)-secure in certain adversarial model if it is both (t_f,ε)-unforgeable and robust in that model.

Two distribution ensembles $\{X_n\}$ and $\{Y_n\}$ are (computationally) *indistinguishable* if for any probabilistic polynomial-time distinguisher D and any polynomial $p(n)$, there is an integer n_0 such that for any $n \geqslant n_0$,

$$|\Pr[D(X_n)=1]-\Pr[D(Y_n)=1]|<1/p(n).$$

The *discrete logarithm* over a safe prime problem is to solve $D\log_g h \bmod p$ from given (p,g,h), where $p=2p'+1$ is prime, p' is also prime, and g is a generator of the quadratic subgroup $G_{p'}$ of Z_p^*. We assume that no probabilistic polynomial-time Turing machine can solve a significant portion of the input. Let I_n be the (uniform) distribution of the size-n legal input. Then, for any probabilistic polynomial-time Turing A and polynomial $p(n)$, there is n_0 such that for any $n \geqslant n_0$,

$$\Pr_{p,g,h\in I_n}[A(p,g,h)=D\log_g h \bmod p]<1/p(n).$$

2.4. The models

2.4.1. Players and adversary model

Our system consists of l players and an honest dealer who is responsible for generating keys and distributing the shares. The adaptive adversary is modeled as a probabilistic polynomial-time Turing machine and may corrupt players at any time during the execution of the protocol. A player is *corrupted* if it is controlled by the adversary. Once a player is corrupted, it may deviate from the protocol in any way, and its secret information is revealed.

2.4.2. Communication model

We assume that the players have access to a round-based and fully synchronous broadcast channel, which can be implemented by a Byzantine agreement protocol [33,31]. If all players broadcast messages simultaneously in a round, they can see all messages from others without delay and alternation. In addition, each pair of players are connected by a private channel such that other players cannot tape. The private channel can be simply implemented by a symmetric or public-key encryption scheme. However, as mentioned in [7], the adaptive security cannot be maintained via such implementation because the plaintexts are ''committed'' by the ciphertexts. If the adversary corrupts some party latter, it can decrypt the ciphertexts received by that party, and check whether the plaintexts are consistent with the values given by the simulator. Since the simulator needs to adjust some inside values of the parties, the simulation will fail. So we use the technique introduced in [3] to implement the private channel, as adopted by [7]. Each party uses a pseudorandom generator to refresh the encryption keys so that the adversary gets no information about the prior keys at the time of corruption.

3. Threshold GQ signature scheme

In our threshold GQ signature scheme, the dealer generates a public/secret key pair and distributes the shares of the secret key to the players. To sign a message distributively, each player produces a partial signature. If there are more than $t+1\left(t<\frac{l}{3}\right)$ valid partial signatures, we can construct the signature of the message from the valid partial signatures.

3.1. Generating keys

The *key generation* process is shown in Fig. 1. Let $\{P_i\}$ be the set of l participating players and $L=l!$ There are two security parameters k_1 and k_2. The dealer chooses two safe primes $p=2p'+1$ and $q=2q'+1$, each of

Input: security parameters k_1, k_2.

The dealer generates and distributes keys as follows:

(1) Choose two $\lceil k_1/2 \rceil$-bit random primes p, q such that $p = 2p' + 1, q = 2q' + 1$, where p', q' are also primes. Let $n = pq, m = p'q'$, and g a generator of Q_n.

(2) Choose a random polynomial $f(x) = a_0 + a_1 x + \ldots + a_t x^t$ over Z_m of degree t. Let $s = g^{a_0} \bmod n$ be the main secret and hand $s_i = g^{f(i)} \bmod n$ to player P_i secretly.

(3) Randomly choose a (k_2+1)-bit value e, such that $\gcd(e, \phi(n)) = 1$ and $\gcd(e, L^2) = 1$, where $L = l!$. Compute $v = 1/s^e \bmod n$.

(4) Let $SK_i = (n, e, g, s_i)$ be the secret key of player P_i, and broadcast the public key $PK = (n, e, g, v)$.

Fig. 1. TH-GQ-KeyGen: Generating keys.

length $\lceil k_1/2 \rceil$ bits, where p' and q' are also primes. Let $n = pq$, $m = p'q'$, Q_n the set of all quadratic residues modulo n, and g a generator of Q_n. The order of Q_n is m. Hereafter, all group computations are done in Q_n and the corresponding exponent arithmetic is done in Z_m.

The dealer then chooses a random degree-t polynomial $f(x)$ over Z_m and gives the share $s_i = g^{f(i)}$ to the player P_i. Note that the share given to player P_i is $g^{f(i)}$, instead of $f(i)$. The shared secret key is thus $s = g^{f(0)}$. The dealer then chooses a random $(k_2 + l)$-bit value e with $\gcd(e, \phi(n)) = \gcd(e, L^2) = 1$ and computes $v = l/s^e \bmod n$. In summary, the public key PK of the scheme is (n, e, g, v) and the secret share of player P_i is $SK_i = (n, e, g, S_i)$.

3.2. Signing messages

Our signing protocol mainly follows the signing steps of GQ signature scheme, except that the values r and y are jointly generated. It consists of two phases: *partial signature generation and signature construction*, shown in Fig. 2.

3.2.1. Partial signature generation

To sign a message M, players jointly compute $y = r^e \bmod n$ first, where r is a random secret value. For the simplicity of partial signature construction, we use $g^{f_r(x)}$ instead of $f_r(x)$ to share the value r. That is, each player p_i gets a share $r_i = g^{f_r(i)} \bmod n$ and r is defined as $g^{f_r(0)}$. Therefore, we provide a protocol INT-JOINT-EXP-RVSS for joint verifiable secret sharing of a random secret on exponent over integers, which is shown in Appendix B. It lets all players jointly compute $y = r^e = g^{f_r(0)e}$, and each player P_i get his own share $g^{f_r(i)}$. Thus $\sigma = H(y, M)$ can be easily computed by all players, where H is the chosen one-way hash function.

All players then jointly execute INT-JOINT-ZVSS (joint verifiable secret sharing of zero), which is just like INT-JOINT-RVSS described in Appendix A except that each player sets his secret a_{i0}, b_{i0} to be zero, to share a zero-constant t-degree polynomial $f_c(x)$. Each player P_i also holds a share $c_i = g^{Lf_c(i)} \bmod n$. This randomized polynomial is generated for the security proof, described in Section 3.4. All other secret information generated in INT-JOINT-ZVSS are then erased (the erasing technique [6,7]). Finally, each player P_i computes his partial signature $Z_i = (r_i S_i^{\sigma})^L C_i \bmod n$.

3.2.2. Signature construction

To compute the signature for M, we need $t + 1$ valid partial signatures. Since we do not know the validity of partial signatures, we have to find the valid ones by simple majority vote or some more efficient algorithms such as Berlekamp–Welch algorithm [38]. Therefore the threshold t must be less than $\frac{l}{3}$. (We discuss how to

Signing message

Input: message M;

Partial signature generation:

1. All players perform INT-JOINT-EXP-RVSS to generate $y = g^{f_r(0)e} \bmod n$ such that each player P_i gets his share $f_r(i)$ and computes $r_i = g^{f_r(i)} \bmod n$.
2. All players compute $\sigma = H(y, M)$.
3. All players perform INT-JOINT-ZVSS such that each player P_i gets a share $f_c(i)$ and computes $c_i = g^{Lf_c(i)} \bmod n$. All secret information, except c_i, generated in this step is erased.
4. Each player P_i computes his partial signature $z_i = (r_i s_i^{\sigma})^L c_i \bmod n$.

Signature construction:

5. Find $t+1$ valid partial signatures $z_{i_1}, z_{i_2}, \ldots, z_{i_{t+1}}$.
6. Compute
$$z' = \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{0,i_j} L} \bmod n,$$
where λ_{0,i_j}'s are the corresponding interpolation coefficients.
7. Find integers a, b for $L^2 a + eb = 1$, and compute $z = z'^a \cdot (y/v^{\sigma})^b \bmod n$.
8. The signature of message M is (z, σ).

Verifying signature

Input: (PK, M, SIG) where $PK = (n, e, g, v), SIG = (z, \sigma)$.

(1) Compute $y' = z^e v^{\sigma} \bmod n$.

(2) Accept the signature if $\sigma = H(y', M)$.

Fig. 2. TH-GQ-Sig and TH-GQ-Ver: Signing and verifying message.

achieve $t < \frac{l}{2}$ in the next section.) Assume that the $t+1$ valid partial signatures are $z_{i_1}, z_{i_2}, \ldots, z_{i_{t+1}}$, we compute the interpolation

$$z' = \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{0,i_j} L} \bmod n = \left(g^{\sum_{j=1}^{t+1} \lambda_{0,i_j} f_r(i_j) + \sigma f(i_j) + f_c(i_j)} \right)^{L^2} \bmod n = (rs^{\sigma})^{L^2} \bmod n,$$

where λ_{0,i_j} is the interpolation coefficient for the constant term from the index set $\{i_1, i_2, \ldots, i_{t+i}\}$. Since $\lambda_{0,i_j} \cdot L$ is an integer, we can compute z' without knowing the factorization of n (and thus m). Moreover, because that $\gcd(L^2, e) = 1$, we can find integers a, b such that $L^2 a + eb = 1$ and compute the signature z for M as:

$$z = z'^a \cdot (y/v^{\sigma})^b = (rs^{\sigma})^{L^2 a} (rs^{\sigma})^{eb} = rs^{\sigma} \bmod n$$

Remark. Since the sharing polynomials $f(x)$, $f_r(x)$, and $f_c(x)$ are over the exponents, the partial signature $z_i = (r_i s_i^{\sigma})^L c_i$ is a share of a degree-t polynomial in the exponent. Thus, we need only $t+1$ shares to interpolate z'. This helps us to modify the protocol to achieve optimal resilience. The detail is described in Section 4.

3.3. Verifying signatures

The verification procedure is straightforward, as defined by the GQ signature scheme, shown in Fig. 2.

3.4. Security analysis

Our threshold GQ signature scheme is secure against the chosen message attack in the adaptive adversary model. As long as the adaptive adversary controls less than $t+1$ players, it cannot forge a valid signature without interacting with un-corrupted players. The adversary cannot interrupt the un-corrupted players to cooperatively obtain a valid signature for a message, either.

We need a simulator $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$ to simulate the view of execution of the TH-GQ-Sig scheme producing a signature (σ, z) for a message M. The simulator is shown in Fig. 3.

Lemma 1. *If the adaptive adversary corrupts at most $t < \frac{l}{3}$ players, its view of an execution of* TH-GQ-Sig *on input message M and output signature (σ, z) is the same as the view of an execution of* $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$ *on input M and signature (σ, z).*

Proof. Assume that $\mathscr{B}$ is the set of corrupted players and $\mathscr{G}$ is the set of un-corrupted (honest) players up to now. In the beginning (the key generation stage), the simulator emulates the dealer to randomly assign $s_i \in Q_n$ to player P_i, $1 \leqslant i \leqslant l$. These s_i's remain fixed for many rounds of simulation of TH-GQ-Sig. Let $y = z^e v^\sigma \bmod n$, which is the correct $r^e \bmod n$. Since y is distributively computed by all players in TH-GQ-Sig, the simulator runs $\mathsf{SIM}_{\mathsf{INT\text{-}JOINT\text{-}EXP\text{-}RVSS}}$ (Fig. B.2) on input y to simulate the execution of INT-JOINT-EXP-RVSS protocol. In Step 3, the simulator runs INT-JOIN-ZVSS on behalf of honest players and assigns each corrupted player P_i a share $c_i = g^{f_c(i)} \bmod n$, where $f_c(x)$ has a zero constant. Now, the corrupted players P_i get s_i, r_i and c_i. Their partial signatures $z_i = (r_i s_i^\sigma)^L c_i \bmod n$ need be fixed since the adversary corrupts them. Let $\Lambda' \supseteq \mathscr{B}$ be a set of t players. We fix the partial signatures of the players in Λ'. For un-corrupted players $P_j \notin \Lambda'$, we set their partial signatures to be compatible with those in Λ', that is, the shares of any $t+1$ players result in the same signature. This is done by setting their partial signatures as

Input: message M and the corresponding signature (σ, z)

Let $\mathcal{B}$ be the set of corrupted players and $\mathcal{G}$ the set of honest players at that time.

Randomly choose $s_i \in Q_n$ for P_i, $1 \leq i \leq l$.

(1) Let $y = z^e v^\sigma \bmod n$.

(2) Execute $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ on input y.

(3) Execute INT-JOINT-ZVSS and assign each corrupted player P_i a share $c_i = g^{f_c(i)L} \bmod n$.

(4) Randomly select a set $\Lambda' \supseteq \mathcal{B}$ of t players and for each $P_j \notin \Lambda'$, compute

$$z_j^* = z^{\lambda_{j,0} L} \cdot \prod_{k \in \Lambda'} (r_k s_k^\sigma g^{f_c(k)})^{\lambda_{j,k} L} \bmod n,$$

set

$$c_j^* = z_j^* / (r_j s_j^\sigma)^L \bmod n,$$

and erase c_j.

(5) Broadcast all partial signatures z_i^* for $i \in \mathcal{G}$. (Note that $z_i^* = z_i$ for $P_i \in \Lambda' - \mathcal{B}$.)

Fig. 3. $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$: Simulator of TH-GQ-Sig.

$$z_j^* = z^{\lambda_{j,0}L} \cdot \prod_{k\in\Lambda'} (r_k s_k^{\sigma} g^{f_c(k)})^{\lambda_{j,k}L} \bmod n,$$

where $\lambda_{j,k}$'s are the interpolation coefficients for computing the jth share from the set of shares $\{0\} \cup \{k|P_k \in \Lambda'\}$. The simulator also sets the new shares $c_j^* = z_j^*/(r_j s_j^{\sigma})^L$ the players $P_j \notin \Lambda'$ and erases the old shares c_j. These c_j^* make the un-corrupted players have the consistent relation for r_j, s_j, c_j^* and z_j^*.

We see how the simulator produces an indistinguishable distribution for the adversary:

(1) The simulator runs $\mathsf{SIM}_{\mathsf{INT\text{-}JOINT\text{-}EXP\text{-}RVSS}}$ which generates y in the proper distribution.
(2) The simulator performs INT-JOINT-ZVSS on behalf of honest players. This is the same as what TH-GQ-Sig does in Step 3. Thus, the distribution for c_i's is the same.
(3) The partial signatures z_i's of all players are consistent since the shares of any $t+1$ players produce the right signature (σ, z) (by adjusting c_i's). Therefore, they have the right distribution.
(4) The erasing technique (for c_i's) is employed. As long as the simulated distribution is indistinguishable for the adversary after it corrupts a new player, the entire distribution is indistinguishable for the adversary after it corrupts up to t players. There is no inconsistency problem between corrupted players.
(5) The shares c_j's are adjusted to c_j^*'s for the un-corrupted players (up to now) so that even the adversary corrupts it later, the partial signature z_j^* is consistent with the possible check of equation $z_j^* = (r_j s_j^{\sigma})^L c_j^*$.

In conclusion, the simulator $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$ produces an indistinguishable distribution for the adversary, who corrupts up to t players in an adaptive way. □

We now show that our threshold GQ signature scheme is secure against the adaptive adversary under the chosen message attack.

For unforgeability, let $\mathcal{O}_{\text{sig}}$ be the signing oracle that a forger (in the centralized version) queries for signatures of messages. When $\mathcal{O}_{\text{sig}}$ returns (σ, z) for a message M, the simulator, on input M and (σ, z), outputs a transcript with an indistinguishable distribution for the adaptive adversary (in the distributed version). Thus, the adversary, who engaged several executions of the TH-GQ-Sig protocol, cannot produce an additional valid signature without cooperation of un-corrupted players.

Theorem 1. *If the underlying GQ signature scheme is (t_f, ε)-secure, the (t, l)-threshold GQ signature scheme in Figs. 1 and 2 is (t'_f, ε')-unforgeable against the adaptive adversary who corrupts up to $t < \frac{l}{3}$ players, where $t'_f \geqslant t_f - q_s t_1(k, l)$, $\varepsilon' \leqslant \varepsilon + q_s/3^k$, t_1 is a time-bounded function, q_s is the number of times of signature queries, and k is the security parameter.*

Proof. Assume that the adversary $\mathscr{A}$, who controls up to t players during execution of the TH-GQ-Sig scheme and thus obtains signatures for $M_1, \ldots, M_{q_s}$, produces a valid signature for M, $M \neq M_i$. We construct a forger $\mathscr{F}$ to forge a signature of the underlying GQ signature scheme for an un-queried message using the procedure $\mathscr{A}$ and the signing oracle $\mathcal{O}_{\text{sig}}$ for the underlying GQ signature scheme.

Let (n, e, g, v) be the public key of the underlying GQ signature scheme. This is used in the (simulated) threshold GQ signature scheme also. First, since $\mathscr{F}$ does not know the corresponding secret key, in the key generation stage $\mathscr{F}$ assigns each player P_i a random secret share $s_i \in Q_n$. Then, it simulates all players and the adversary $\mathscr{A}$. When the adversary $\mathscr{A}$ executes TH-GQ-Sig to produce a valid signature for M_i, $1 \leqslant i \leqslant q_s$, $\mathscr{F}$ queries $\mathcal{O}_{\text{sig}}$ to obtain a signature (σ_i, z_i) and runs the simulator $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$, on input M_i and (σ_i, z_i), to produce a transcript T_i with right distribution (by Lemma 1) for $\mathscr{A}$. Therefore, $\mathscr{F}$ simulates $\mathscr{A}$, on input of these transcripts T_i's, to produce a valid signature (σ, z) for a new message M, $M \neq M_i$. Thus the underlying GQ signature is not unforgeable under the chosen message attack, which is a contradiction.

While $\mathscr{F}$ performs $\mathsf{SIM}_{\mathsf{INT\text{-}JOINT\text{-}EXP\text{-}RVSS}}$ (Fig. B.2) in the $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$, if the player P_u is corrupted by the adversary, the simulator must rewind the simulation and pick another single-inconsistent-player. The probability of this inconsistent player being corrupted is at most 1/3 (this is given by the ratio t/l and $t < \frac{l}{3}$). Therefore, $\mathsf{SIM}_{\mathsf{TH\text{-}GQ\text{-}Sig}}$ would fail with probability at most $q_s/3^k$ if the simulator rewinds at most k times in one of q_s times signature queries. The time complexity of the forger $\mathscr{F}$ is $t'_f + q_s t_1(k, l)$ while t_1 is the additional rewinding time complexity function of $\mathsf{SIM}_{\mathsf{INT\text{-}JOINT\text{-}EXP\text{-}RVSS}}$. SO if the underlying GQ signature scheme is

(t_f, ε)-secure, our threshold GQ signature scheme is (t'_f, ε')-unforgeable, where $t'_f \geqslant t_f - q_s t_1(k,l)$ and $\varepsilon' \leqslant \varepsilon + q_s/3^k$. □

Theorem 2. *If computing the discrete logarithm modulo a safe prime is hard, the* TH-GQ-Sig *scheme in Fig. 2 is robust against the adaptive adversary.*

Proof. If there is an adversary $\mathscr{A}'$ who participates TH-GQ-Sig on the input messages that it selected such that the honest players fail to generate a valid signature for a given message, then we can construct an extractor $\mathscr{E}$ to break the discrete-log problem modulo a safe prime. That is, on input $(\tilde{p}, \tilde{g}, \tilde{h})$, $\mathscr{E}$ can compute $D\log_{\tilde{g}}\tilde{h} \bmod \tilde{p}$ where $\tilde{p} = 2\tilde{p}' + 1, \tilde{g}, \tilde{h}$ are generators of $G_{\tilde{p}'}$.

First, $\mathscr{E}$ lets the dealer generate the related keys as usual (Fig. 1) except that the dealer chooses (1) $p = \tilde{p}$ (and thus $p' = \tilde{p}'$) (2) $g = \tilde{g}^2 \bmod n$. Note that the probability of $g \equiv 1 \pmod q$ is only $1/q$. Since g is the generator of both $G_{p'}$ and $G_{q'}$, g is a generator of Q_n. For another generator $\tilde{h}$, $\mathscr{E}$ simulates h-generation protocol [27] with $\mathscr{A}'$ and outputs $h = \tilde{h}$. Using the instance (g,h), $\mathscr{E}$ performs TH-GQ-Sig with $\mathscr{A}'$ on behalf of the honest players. Now, we show that if $\mathscr{A}'$ hinders the signing protocol from producing a valid signature, $\mathscr{E}$ can compute $\mathscr{D} = D\log_g h \bmod n$, and then outputs $D\log_{\tilde{g}}\tilde{h} = 2\mathscr{D} \bmod \tilde{p}$.

Let us consider where the protocol may fail to produce the valid signature. First, in executing the INT-JOINT-RVSS scheme (or INT-JOINT-ZVSS, see Fig. A.1), if a corrupted player P_i distributes his shares $(f_i(j), f'_i(j))$, $1 \leqslant j \leqslant n$, that pass the verification equation (A.1), but do not lie on a t-degree polynomial, the extractor $\mathscr{E}$ can solve the system of $t+2$ linearly independent equations of the form $c_0 + c_1 j + c_2 j^2 + \cdots + c_t j^t = f_i(j) + \mathscr{D} f'_i(j)$ with $t+2$ unknown variables $c_0, c_1, \ldots, c_t$ and $\mathscr{D}$. Then, the extractor outputs $\mathscr{D}$.

Another situation that $\mathscr{A}'$ may cheat is on the zero-knowledge proof in executing INT-JOINT-EXP-RVSS (Fig. B.1). If the corrupted player P_i broadcasts $(A_i^*, B_i^*) \neq (g^{a_{i0}e}, h^{b_{i0}e})$ in Step 2, $\mathscr{E}$ extracts $\mathscr{D} = D\log_g h$ as follows. Assume that $A_i^* = g^{a'} \bmod n$ and $B_i^* = h^{b'} \bmod n$. After executing Steps 2a–2c, $\mathscr{E}$ gets $R_i = r_i + da'e$ and $R'_i = r'_i + d^{b'}e$. Then $\mathscr{E}$ rewinds $\mathscr{A}'$ to run Steps 2b–2c again. This gives $\mathscr{E}$ another two equations $R_i^* = r_i + d^* a' e$ and $R_i'^* = r'_i + d^* b' e$. As long as $d \neq d^*$ (the probability of equality is negligible), $\mathscr{E}$ can solve the equations and get the four unknown variables r_i, r'_i, a', b'. Since $\mathscr{E}$ knows the value m, the extractor computes $\mathscr{D}$ from $a_{i0} + \mathscr{D} b_{i0} = a' + \mathscr{D} b' \bmod m$. □

4. Achieving optimal resilience

The protocols presented up to now have not yet achieved optimal resilience since in Step 5 of TH-GQ-Sig (Fig. 2), we have to find $t+1$ valid partial signatures. Also, the Step 2(b) of INT-JOINT-EXP-RVSS (Fig. B.1) needs to correctly reconstruct the challenge d. Without validity proof of these shares, the threshold t cannot be more than $\frac{l}{3}$. Moreover, because all players generate secrets jointly, the number of honest players must be more than that of dishonest ones. We describe how to modify them to achieve optimal resilience ($n \geqslant 2t+1$).

4.1. The optimal resilient scheme

The main difference of this modified scheme is that each player proves the correctness of its partial signature when issued. For this proof, in the key generation stage the dealer directly shares $f(i)$ (instead of $g^{f(i)}$) to player P_i. In addition, the dealer shares another random polynomial $f'(x)$ and broadcasts the public references to $f(i)$ and $f'(i)$ for each P_i in the unconditionally-secure form, that is, the value $S_i = g^{f(i)} h^{f'(i)}$ where h is another generator of Q_n. The main key share of P_i is still set to $s_i = g^{f(i)}$, and all other operations are the same. The whole process OR-TH-GQ-KeyGen is shown in Fig. 4.

In the partial signature generation phase, each P_i gets the additional $f'_r(i)$ and $f'_c(i)$ from INT-JOINT-EXP-RVSS and INT-JOINT-ZVSS, respectively. The unconditionally-secure references $R_i = g^{fr(i)} h^{f'_r(i)}$ and $C_i = g^{Lf_c(i)} h^{Lf'_c(i)}$ can be also computed in the corresponding protocols. Now, each P_i knows the secrets $f(i)$, $f_r(i)$, and $f_c(i)$, and the corresponding reference information are publicly known. When signing a partial signature, each player presents a non-interactive zero-knowledge proof of knowledge of the secret shares. The proof is described as follows:

Input: security parameters k_1, k_2.

The dealer generates and distributes keys as follows:

(1) Choose two $\lceil k_1/2 \rceil$-bit random primes p, q such that $p = 2p' + 1, q = 2q' + 1$, where p', q' are also primes. Let $n = pq, m = p'q'$, and g, h are generators of Q_n.

(2) Choose two random polynomials $f(x) = a_0 + a_1 x + \ldots + a_t x^t$ and $f'(x) = a'_0 + a'_1 x + \ldots + a'_t x^t$ over Z_m of degree t. Let $s = g^{a_0} \bmod n$ be the main secret and hand $f(i), f'(i)$ to player P_i secretly.

(3) Randomly choose a (k_2+1)-bit value e, such that $\gcd(e, \phi(n)) = 1$ and $\gcd(e, L^2) = 1$, where $L = l!$. Compute $v = 1/s^e \bmod n$.

(4) Let $s_i = g^{f(i)}, s'_i = h^{f'(i)}$, and $SK_i = (n, e, g, s_i, s'_i)$ be the secret key of P_i.

(5) Let $S_i = g^{f(i)} h^{f'(i)}$, and broadcast the public key $PK = (n, e, g, v)$ and the partial signature verification keys $S_1, S_2, \cdots, S_l$.

Fig. 4. OR-TH-GQ-KeyGen: Generating keys.

P proves to V the knowledge of x and r committed by $E = g^x h^r$.
Let H_1 be a cryptographic strong hash function.

(1) P chooses two random values $w, w' \in Z_n^*$.
(2) P computes $\gamma = H_1(g \| h \| E \| g^w h^{w'})$, and $D = w + \gamma x$, $D' = w' + \gamma r$.
(3) P sends (γ, D, D') to V.
(4) V checks whether $\gamma = H_1(g \| h \| E \| g^D h^{D'} E^{-\gamma})$.

We let $E = (R_i S_i^{\sigma})^L C_i \bmod n$ be the commitment of $(f_r(i) + f(i)\sigma)L + f_c(i)$, and each P_i can prove the knowledge of them. Therefore each P_i generates a partial signature $\mathscr{PS}_i = (z_i, z'_i, \gamma_i, D_i, D'_i)$ instead of z_i. When constructing the signature, all players first check that $(R_i S_i^{\sigma})^L C_i \equiv z_i z'_i \bmod n$ to ensure that z_i and z'_i are correct with respect to the reference value. Then the verification of the non-interactive proof makes sure that P_i knows the committed value. After that, we can pick up exactly $t + 1$ correct partial signatures to reconstruct the signature of M. The modified signing protocol OR-TH-GQ-Sig is described in Fig. 5.

For the case of Step 2(b) of INT-JOINT-EXP-RVSS in Fig. B.1, since all players perform INT-JOINT-RVSS to generate their shares, the verification equation (1) in Fig. A.1 will sieve out the bad shares. Therefore, the reconstruction of d in INT-JOINT-EXP-RVSS achieves optimal resilience.

4.2. Security analysis

We prove that our optimal resilient threshold GQ signature scheme described above is secure against the chosen message attack in the adaptive adversary and random oracle model. As in the previous proof, we need to check both unforgeability and robustness properties. First, we construct a simulator $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ to simulate the adversarial view of the execution of OR-TH-GQ-Sig, and get the following lemma.

Lemma 2. *If the adaptive adversary corrupts at most $t < \frac{l}{2}$ players, its view of an execution of* OR-TH-GQ-Sig *on input message M and output signature (σ, z) is the same as the view of an execution of* $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ *on input M and signature (σ, z).*

Proof. The simulation steps of $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ are the same as $\mathsf{SIM}_{\text{TH-GQ-Sig}}$, except that we need to simulate the additional view of R_i, C_i, z'_i, γ_i, D_i, and D'_i for each $i \in \mathscr{G}$. Since the simulation performs INT-JOINT-RVSS (or INT-JOINT-ZVSS) as the usual execution, the reference values R_i's, C_i's have the right distribution.

Signing message

Input: message M.

Partial signature generation:

1. All players perform INT-JOINT-EXP-RVSS to generate $y = g^{f_r(0)e} \bmod n$ such that each player P_i gets his shares $f_r(i), f'_r(i)$ and computes $r_i = g^{f_r(i)}, r'_i = h^{f'_r(i)} \bmod n$. The value $R_i = g^{f_r(i)} h^{f'_r(i)} \bmod n$ can be publicly computed by the committed values in INT-JOINT-EXP-RVSS.
2. All players compute $\sigma = H(y, M)$.
3. All players perform INT-JOINT-ZVSS such that each player P_i gets shares $f_c(i), f'_c(i)$ and computes $c_i = g^{Lf_c(i)}, c'_i = h^{Lf'_c(i)} \bmod n$. The value $C_i = g^{Lf_c(i)} h^{Lf'_c(i)} \bmod n$ can be publicly computed by the committed values in INT-JOINT-ZVSS. All secret information, except c_i, c'_i, generated in this step is erased.
4. Each player P_i computes his partial signature $\mathcal{PS}_i = (z_i, z'_i, \gamma_i, D_i, D'_i)$ where
   - $z_i = (r_i s_i^{\sigma})^L c_i \bmod n$,
   - $z'_i = (r'_i s'^{\sigma}_i)^L c'_i \bmod n$,
   - $w_i, w'_i \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$,
   - $VK_i = (R_i S_i^{\sigma})^L C_i \bmod n$,
   - $\gamma_i = H_1(g||h||VK_i||g^{w_i} h^{w'_i} \bmod n)$,
   - $D_i = w_i + \gamma_i((f_r(i) + f(i)\sigma)L + f_c(i))$,
   - $D'_i = w'_i + \gamma_i((f'_r(i) + f'(i)\sigma)L + f'_c(i))$.

   The values w_i and w'_i are erased. P_i outputs the partial signature $\mathcal{PS}_i$.

Signature construction:

1. Verify each partial signature $\mathcal{PS}_i = (z_i, z'_i, \gamma_i, D_i, D'_i)$ by checking if the following conditions are satisfied:
   - $(R_i S_i^{\sigma})^L C_i \equiv z_i z'_i \bmod n$,
   - $\gamma_i = H_1(g||h||z_i z'_i \bmod n||g^{D_i} h^{D'_i} (z_i z'_i)^{-\gamma_i} \bmod n)$.
2. Compute
$$z' = \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{0,i_j} L} \bmod n,$$
   where $z_{i_1}, z_{i_2}, \ldots, z_{i_{t+1}}$ are $t+1$ partial signatures passing the above verification and λ_{0,i_j}'s are the corresponding interpolation coefficients.
3. Find integers a, b for $L^2 a + eb = 1$, and compute $z = z'^a \cdot (y/v^{\sigma})^b \bmod n$.
4. The signature of message M is (z, σ).

Fig. 5. OR-TH-GQ-Sig: Signing a message.

Moreover, the partial signatures z_i's can be perfectly simulated in the Step 4 of $\mathsf{SIM}_{\text{TH-GQ-Sig}}$. So we can simulate the values $z'_i = (R_i S_i^{\sigma})^L C_i / z_i \bmod n$ for each $i \in \mathcal{G}$ s.t. these z'_i's are all correctly distributed.

For the non-interactive proof, we need a random oracle to simulate the hash function H_1. The simulator first randomly chooses γ_i, D_i, and D'_i for each $i \in \mathcal{G}$, and adds $(g||h||z_i z'_i \bmod n||g^{D_i} h^{D'_i} (z_i z'_i)^{-\gamma_i} \bmod n, \gamma_i)$ to a list

L_{H_1}. Then $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ need not choose w_i and w_i' because these values are erased before outputting the partial signature. If there is a query Q^* made to H_1, $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ returns the corresponding γ^* if Q^* is on the list L_{H_1}, or returns a random value otherwise. Therefore, the values γ_i's, D_i's, and D_i''s are all in the proper distribution and would pass the verification of partial signatures. □

Because of the successful simulation, we have the same unforgeability property as TH-GQ-Sig.

Theorem 3. *If the underlying GQ signature scheme is* (t_F, ϵ)*-secure, the optimal resilient* (t, l)*-threshold GQ signature scheme in Figs. 4 and 5 is* (t_f', ε')*-unforgeable against the adaptive adversary who corrupts up to* $t < \frac{1}{2}$ *players, where* $t_f' \geqslant t_f - q_s t_1(k, l)$, $\varepsilon' \leqslant \varepsilon + g_s/2^k$, t_1 *is a time-bounded function,* g_s *is the number of times of signature queries, and* k *is the security parameter.*

Proof. The proof is the same as the proof of Theorem 1 except that we replace $\mathsf{SIM}_{\text{TH-GQ-Sig}}$ with $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ to simulate the execution of OR-TH-GQ-Sig. By Lemma 2, $\mathsf{SIM}_{\text{OR-TH-GQ-Sig}}$ outputs the transcripts with right distribution. So the theorem holds. □

For the robustness property, we get the same argument as Theorem 2.

Generating Keys

Input: security parameters (k_1, k_2, T).

The dealer generates and distributes keys as follows:

(1) Choose two $\lceil k_1/2 \rceil$-bit random primes p, q such that $p = 2p' + 1, q = 2q' + 1$, where p', q' are also primes. Let $n = pq, m = p'q'$, and g a generator of Q_n.

(2) Choose a random polynomial $f(x) = a_0 + a_1 x + \ldots + a_t x^t$ over Z_m of degree t. Let $s = g^{a_0} \bmod n$ be the main secret and hand $s_i = g^{f(i)} \bmod n$ to player P_i secretly.

(3) For all $j = 1, \ldots, T$, choose a prime e_j such that $2^{k_2}(1+(j-1)/T) \leq e_j < 2^{k_2}(1+j/T)$.

(4) Compute the public value $v = 1/s^{e_1 \cdot \ldots \cdot e_T} \bmod n$.

(5) Compute the first period share $s_{i1} = s_i^{e_2 \cdot \ldots \cdot e_T} \bmod n$ and $t_{i1} = s_i^{e_1} \bmod n$ for each player P_i.

(6) Send the secret share $SK_{i1} = (1, T, n, e_1, g, s_{i1}, t_{i1})$ to each player P_i at time period 1 and publish the public key $PK = (n, g, v, T)$.

Updating Keys

Input: $SK_{ij} = (j, T, n, e_j, g, s_{ij}, t_{ij})$ for each player P_i.

Each player P_i updates his share from time period j to $j + 1$.

(1) Each player regenerates $e_{j+1}, \ldots, e_T$.

(2) Each player P_i updates his share by computing:

$$s_{i(j+1)} = t_{ij}^{e_{j+2} \cdot \ldots \cdot e_T} \bmod n; \quad t_{i(j+1)} = t_{ij}^{e_{j+1}} \bmod n$$

(3) The new secret share of player P_i at time period $j + 1$ is $SK_{i(j+1)} = (j + 1, T, n, s_{i(j+1)}, t_{i(j+1)}, e_{j+1})$.

Fig. 6. TH-FS-GQ: Generating and updating keys.

Signing Message

Input: message M

For time period j, players jointly sign the message as follows:

Partial signature generation:

1. All players perform INT-JOINT-EXP-RVSS to generate $y = g^{f_r(0)e_j} \bmod n$ such that each player P_i gets his share $f_r(i)$ and computes $r_i = g^{f_r(i)} \bmod n$.
2. All players compute $\sigma = H(j, e_j, y, M)$.
3. All players perform INT-JOINT-ZVSS such that each player P_i gets a share $f_c(i)$ and computes $c_i = g^{Lf_c(i)} \bmod n$. All secret information, except c_i, generated in this step is erased.
4. Each player P_i computes his partial signature $z_i = (r_i s_{i,j}^{\sigma})^L c_i \bmod n$.

Signature construction:

5. Find $t+1$ valid partial signatures $z_{i_1}, z_{i_2}, \ldots, z_{i_{t+1}}$.
6. Compute
$$z' = \prod_{k=1}^{t+1} z_{i_k}^{\lambda_{0,i_k} L} \bmod n,$$
where λ_{0,i_k}'s are the corresponding interpolation coefficients.
7. Find integers a, b for $L^2 a + e_j b = 1$, and compute $z = z'^a \cdot (y/v^{\sigma})^b \bmod n$.
8. The signature of message M is (z, σ, j, e_j).

Verifying Signature

Input: (PK, M, SIG, k_2) where $PK = (n, g, v, T), SIG = (z, \sigma, j, e')$, k_2 is the security parameter.

The verifier accepts the signature SIG at time period j if the following equations hold:

(1) $2^{k_2} \le e' < 2^{k_2}(1 + j/T)$.

(2) $\sigma = H(j, e', y', M)$, where $y' = z^{e'} v^{\sigma} \bmod n$.

Fig. 7. TH-FS-GQ: Signing and verifying.

Theorem 4. *If computing the discrete logarithm modulo a safe prime is hard, the* OR-TH-GQ-Sig *scheme in Fig. 5 is robust against the adaptive adversary.*

Proof. The proof steps are exactly the same as the proof of Theorem 2. □

5. Other extensions

5.1. Threshold identity-based signatures

From our threshold GQ signature scheme, we get a threshold *identity-based* signature scheme immediately. The dealer first chooses n, g, e like performing in Fig. 1. For an identity ID, the dealer computes $v_{ID} = H_2(ID)$, where $H_2 : \{0,1\}^* \to Q_n$ is a hash function. Then the main secret s_{ID} is computed as $v_{ID}^{-d} \bmod n$. For each player

P_i, the dealer generates $s_i = g^{f(i)} \bmod n$ such that $g^{f(0)} = s_{ID}$. After generating keys, players can sign messages as usual, and the verifier can verify signatures from the public key $v_{ID} = H_2(ID)$.

5.2. Threshold forward-secure signatures

Based on our threshold GQ signature scheme and Itkis and Reyzin's forward signature scheme [26], we construct a threshold forward-secure signature scheme TH-FS-GQ.

5.2.1. Key generation

The key generation process is in Fig. 6. Let T be the total number of time periods. The first two steps are the same as those of TH-GQ-KeyGen. Each player P_i gets s_i as his initial share. The dealer chooses primes e_j, $1 \leqslant j \leqslant T$, in the range between $2^{k_2}(1 + (j-1)/T)$ and $2^{k_2}(1 + j/T)$. That is, we divide the interval between 2^{k_2} and 2^{k_2+1} into T buckets, and choose each prime e_j from the jth bucket. The public key is $v = 1/s^{e_1 e_2 \ldots e_T} \bmod n$. At time period j, $1 \leqslant j \leqslant T$, the player P_i, $1 \leqslant i \leqslant n$, holds two secrets:

Input: (n, g, h), where n is the product of two large primes and g, h are generators of Q_n.

(1) Each player P_i chooses two random polynomials of degree t:

$$f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t, \quad f'_i(x) = b_{i0} + b_{i1}x + \ldots + b_{it}x^t$$

where

(a) $a_{i0} = L^2\hat{s}, \hat{s} \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$.

(b) $b_{i0} = L^2\hat{s}', \hat{s}' \in_R \{0, \ldots, n^2(\lfloor n/4 \rfloor - 1)\}$.

(c) $a_{ik}, b_{ik} \in_R \{0, L, 2L, \ldots, L^3n^2\}$, $k = 1, \ldots, t$.

(2) Each player P_i broadcasts $C_{ik} = g^{a_{ik}}h^{b_{ik}} \bmod n$, $1 \leq k \leq t$, and hands $f_i(j), f'_i(j)$ to player P_j, $j \in \{1, \ldots, l\}$.

(3) Each player P_j verifies his shares received from each other P_i by checking:

$$g^{f_i(j)}h^{f'_i(j)} \equiv \prod_{k=0}^{t} C_{ik}^{j^k} \bmod n \tag{A.1}$$

If the check fails for an index i, P_j broadcasts a *complaint* message against P_i.

(4) Each player P_i who received a complaint from player P_j broadcasts the corresponding shares $f_i(j), f'_i(j)$.

(5) Each player marks as *disqualified* any player that

- received more than t complaints, or
- answered to a complaint with values that falsify Eq. A.1.

(6) Each player then builds a common set of non-disqualified players $QUAL$. The secret x is now defined as $x = \sum_{i \in QUAL} a_{i0}$, and each player P_i holds a share $x_i = \sum_{j \in QUAL} f_j(i)$.

Fig. A.1. INT-JOINT-RVSS.

Input: (n, e, g, h), where n is the product of two large primes and g, h are generators of Q_n.

(1) Each player P_i performs INT-JOINT-RVSS and gets the following secret outputs:
- Polynomials $f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t, f'_i(x) = b_{i0} + b_{i1}x + \ldots + b_{it}x^t$ he generated randomly.
- The shares $f_j(i), f'_j(i)$ sent from player j, where $j = 1, \ldots, l$.
- The share $f(i) = \sum_{j \in QUAL} f_j(i)$ of the secret x.

The public outputs are $C_{ik} = g^{a_{ik}} h^{b_{ik}} \bmod n$ where $i = 1, \ldots, l$, $k = 0, \ldots, t$, and the set $QUAL$ of non-disqualified players.

(2) Each player $P_i, i \in QUAL$ broadcasts $A_i = g^{a_{i0}e} \bmod n$ and $B_i = h^{b_{i0}e} \bmod n$ such that $A_i B_i = C_{i0}^e$, and prove the knowledge of $a_{i0}e, b_{i0}e$:
   (a) P_i chooses $r_i, r'_i \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and broadcasts $T_i = g^{r_i} \bmod n$, $T'_i = h^{r'_i} \bmod n$.
   (b) All players jointly execute INT-JOINT-RVSS and then publicly reconstruct the random secret d.
   (c) Each player P_i broadcasts $R_i = r_i + d \cdot a_{i0}e$ and $R'_i = r'_i + d \cdot b_{i0}e$.
   (d) Each player P_j checks that $g^{R_i} \equiv T_i \cdot A_i^d \bmod n$ and $h^{R'_i} \equiv T'_i \cdot B_i^d \bmod n$ for $i = 1, \ldots, l$. If the check fails for some index i, P_j complains against P_i.

(3) For each player P_i receives more than t complaints, P_j broadcasts $f_i(j)$. All players reconstruct a_{i0} and compute $A_i = g^{a_{i0}e} \bmod n$.

(4) All players then compute $y = \prod_{j \in QUAL} A_j \bmod n$

Fig. B.1. INT-JOINT-EXP-RVSS.

$$s_{ij} = s_i^{e_1 \cdots e_{j-i} e_{j+1} \cdots e_T} \bmod n \quad \text{and} \quad t_{ij} = s_i^{e_1 e_2 \cdots e_j} \bmod n$$

where s_{ij} is the signing share and t_{ij} is an auxiliary secret for share update. Finally, the dealer sends the player P_i two shares s_{i1} and t_{i1}. Hence, the secret key for player P_i at initial time period is $SK_{i1} = (1, T, n, s_{i1}, t_{i1}, e_1)$ and the system public key is $PK = (n, v, T)$.

5.2.2. Key update

To compute the keys of the $(j+1)$th time period, each player P_i computes the exponents $e_{j+1}, e_{j+2}, \ldots, e_T$ for evolving the secret.[1] Then, P_i computes its signing share $s_{i(j+1)} = t_{ij}^{e_{j+2} e_{j+3} \cdots e_T} \bmod n$ and key evolving secret $t_{i(j+1)} = t_{ij}^{e_{j+1}} \bmod n$. Thus, P_i's signing share in the $(j+1)$th period is

$$SK_{i(j+1)} = (j+1, T, n, s_{i(j+1)}, t_{i(j+1)}, e_{j+1}).$$

5.2.3. Signing messages

The signing protocol, shown in Fig. 7, is quite similar to TH-GQ-Sig except that it involves time parameters and replaces the values e and s_i, with e_j and s_{ij} for time period j. Note that the time period j and the exponent e_j

[1] We can also take all e_j's as the part of the secret key, but this would increase the storage space linear in T. Therefore, we only put e_j into the secret key for each message signing, and regenerate others when updating keys. We assume that the dealer chooses these primes by a deterministic algorithm such that players can regenerate $e_{j+1}, \ldots, e_T$ based on e_j.

are put into the hash function in computing σ. Since the verifier does not know e_j exactly, the hashing ensures that the signer cannot output arbitrary e_j after seeing the challenge σ. Finally, the signature for the message M is (z, σ, j, e_j).

5.2.4. Signature verification

In order to have a constant-size public key, the time period exponents $e_1, e_2, \ldots, e_T$ are not contained in the public key. Since the verifier gets e' from the signature, it needs to make sure that e' is a value between $2^{k_2}(1+(j-1)/T)$ and $2^{k_2}(1+j/T)$. Once e' is verified, the remaining verification is the same as that in TH-GQ-Ver.

6. Conclusion

We have presented the first threshold GQ signature scheme and extended it to achieve optimal resilience in the random oracle model. Our schemes are secure against the adaptive adversary. It is interesting to improve the efficiency of the schemes.

Appendix A. INT-JOINT-RVSS protocol

The INT-JOINT-RVSS protocol allows players jointly generate a random secret, and each player holds a share of the secret. Since we do not know the order of Q_n, the sharing polynomial should be over integers. We can find the unconditionally-secure verifiable secret sharing protocol over integers (INT-VSS) in

Input: the result value y and parameters (n, e, g, h)

(1) Perform INT-JOINT-RVSS on behalf of honest players.

(2) Choose an uncorrupted player P_u, and do the following computation:

- Compute $A_i = g^{a_{i0}e} \bmod n$ and $B_i = h^{b_{i0}e} \bmod n$ for $i \in QUAL\backslash\{u\}$.
- Set $A_u^* = y \cdot \prod_{i \in QUAL\backslash\{u\}} A_i^{-1} \bmod n$ and $B_u^* = C_{u0}^e / A_u^* \bmod n$.
- Choose $d^*, R_u, R'_u \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and set $T_u^* = g^{R_u} \cdot (A_u^*)^{-d^*} \bmod n$, $T_u'^* = g^{R'_u} \cdot (B_u^*)^{-d^*} \bmod n$

(3) Broadcast A_i for player $P_i, i \in QUAL\backslash\{u\}$ and A_u^* for player P_u.

(4) Perform Step 2a in the protocol on behalf of each player $P_i, i \in QUAL\backslash\{u\}$, and broadcast T_u^* and $T_u'^*$ for player P_u.

(5) Perform INT-JOINT-RVSS on behalf of honest players as Step 2b in the protocol, and each P_i gets a share d_i. Pick a random t-degree polynomial $f_d^*(x)$ over integers such that $f_d^*(0) = d^*, f_d^*(i) = d_i$ for $i \in \mathcal{B}$. Erase all other secret information generated in INT-JOINT-RVSS.

(6) Broadcast $f_d^*(i)$ for each honest player P_i.

(7) Broadcast R_i, R'_i computed as Step 2c for player P_i, $i \in \mathcal{G}\backslash\{u\}$. Broadcast $R_u^*, R_u'^*$ for player P_u.

(8) Verify the proof as in the protocol. If players in $\mathcal{B}$ receive more than t complaints, all other players reconstruct their secrets.

(9) Erase all secret information except $f(i)$.

Fig. B.2. $\mathsf{SIM}_{\text{INT-JOINT-EXP-RVSS}}$.

[17,19,21]. So we let each player perform INT-VSS as a dealer to share a random value. After joint execution, the random secret is implicitly defined as the sum of all random values. The protocol is shown in Fig. A.1.

Appendix B. INT-JOINT-EXP-RVSS protocol

Our INT-JOINT-EXP-RVSS (Fig. B.1) is like the adaptively-secure distributed key generation protocol [7] except for the composite modulus and an additional constant exponent. In our protocol, players first jointly perform INT-JOINT-RVSS to share a random secret x. To compute $y = g^{xe} \bmod n$, each player P_i broadcasts $A_i = g^{a_{i0}e}$, $B_i = g^{b_{i0}e}$ where $\sum_{i \in QUAL} a_{i0} = x$, and proves the knowledge of $a_{i0}e, b_{i0}e$ by simultaneous proof technique [7,27].

We also provide a simulator for INT-JOINT-EXP-RVSS in Fig. B.2. On input $y = g^{xe} \bmod n$, the simulator constructs the same adversarial view as in the real protocol running.

References

[1] Michel Abdalla, Sara Miner, Chanathip Namprempre, Forward-secure threshold signature schemes, in: Proceedings of the CT-RSA Conference, Springer-Verlag, 2001, pp. 441–456.

[2] Michel Abdalla, Leonid Reyzin, A new forward-secure digital signature scheme, in: Proceedings of Advances in Cryptology – ASIACRYPT 2000, LNCS, vol. 1976, Springer-Verlag, 2000, pp. 116–129.

[3] Donald Beaver, Stuart Haber, Cryptographic protocols provably secure against dynamic adversaries, in: Proceedings of Advances in Cryptology – EUROCRYPT'92, LNCS, vol. 658, Springer-Verlag, 1992, pp. 307–323.

[4] Mihir Bellare, Sara K. Miner, A forward-secure digital signature scheme, in: Proceedings of Advances in Cryptology – CRYPTO'99, LNCS, vol. 1666, Springer-Verlag, 1999, pp. 431–448.

[5] Jan Camenisch, Maciej Koprowski, Fine-grained forward-secure signature schemes without random oracles, Discrete Appl. Math. 154 (2) (2006) 175–188.

[6] Ran Canetti, Uriel Feige, Oded Goldreich, Moni Naor, Adaptively secure multi-party computation, in: Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96), ACM, 1996, pp. 639–648.

[7] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, Adaptive security for threshold cryptosystems, in: Proceedings of Advances in Cryptology – CRYPTO'99, LNCS, vol. 1666, Springer-Verlag, 1999, pp. 98–115.

[8] Manuel Cerecedo, Tsutomu Matsumoto, Hideki Imai, Efficient and secure multiparty generation of digital signatures based on discrete logarithms, IEICE Trans. Fundamentals E76-A (4) (1993) 532–545.

[9] Ivan Damgård, Kasper Dupont, Efficient threshold RSA signatures with general moduli and no extra assumptions, in: Proceedings of the Public-Key Cryptography (PKC'05), LNCS, vol. 3386, Springer-Verlag, 2005, pp. 346–361.

[10] Ivan Damgård, Maciej Koprowski, Practical threshold RSA signatures without a trusted dealer, in: Proceedings of Advances in Cryptology – EUROCRYPT 2001, LNCS, vol. 2045, Springer-Verlag, 2001, pp. 152–165.

[11] Alfredo De Santis, Yvo Desmedt, Yair Frankel, Moti Yung, How to share a function securely, in: Proceedings of the 26th Annual ACM Symposium on the Theory of Computing (STOC'94), ACM, 1994, pp. 522–533.

[12] Olivier Delos, Jean-Jacques Quisquater, An identity-based signature scheme with bounded life-span, in: Proceedings of Advances in Cryptology – CRYPTO'94, LNCS, vol. 839, Springer-Verlag, 1994, pp. 83–94.

[13] Yvo Desmedt, Society and group oriented cryptography: a new concept, in: Proceedings of Advances in Cryptology – CRYPTO'87, LNCS, vol. 293, Springer-Verlag, 1987, pp. 120–127.

[14] Yvo Desmedt, Yair Frankel, Shared generation of authenticators and signatures, in: Proceedings of Advances in Cryptology – CRYPTO'91, LNCS, vol. 576, Springer-Verlag, 1991, pp. 457–469.

[15] Amos Fiat, Adi Shamir, How to prove yourself: Practical solutions to identification and signature problems, in: Proceedings of Advances in Cryptology – CRYPTO'86, LNCS, vol. 263, Springer-Verlag, 1986, pp. 186–194.

[16] Yair Frankel, Yvo Desmedt, Parallel reliable threshold multisignature, Technical Report TR-92-04-02, Dept. of EE and CS, U. of Winsconsin, April 1992.

[17] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, Moti Yung, Optimal-resilience proactive public-key cryptosystems, in: Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS'97), IEEE, 1997, pp. 384–393.

[18] Yair Frankel, Peter Gemmell, Moti Yung, Witness-based cryptographic program checking and robust function sharing, in: Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96), ACM, 1996, pp. 499–508.

[19] Yair Frankel, Philip D. MacKenzie, Moti Yung, Robust efficient distributed RSA-key generation, in: Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC'98), ACM, 1998, pp. 663–672.

[20] Yair Frankel, Philip D. MacKenzie, Moti Yung, Adaptively-secure distributed public-key systems, in: Proceedings of 7th Annual European Symposium on Algorithms (ESA'99), LNCS, vol. 1643, Springer-Verlag, 1999, pp. 4–27.

[21] Yair Frankel, Philip D. MacKenzie, Moti Yung, Adaptively-secure optimal-resilience proactive RSA, in: Proceedings of Advances in Cryptology – ASIACRYPT'99, LNCS, vol. 1716, Springer-Verlag, 1999, pp. 180–194.

[22] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, Robust and efficient sharing or RSA functions, in: Proceedings of Advances in Cryptology – CRYPTO'96, LNCS, vol. 1109, Springer-Verlag, 1996, pp. 157–172.

[23] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, Robust threshold DSS signatures, in: Proceedings of Advances in Cryptology – EUROCRYPT'96, LNCS, vol. 1070, Springer-Verlag, 1996, pp. 354–371.
[24] Oded Goldreich, Silvio Micali, Avi Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, in: Proceedings of the 19th Annual ACM Symposium on the Theory of Computing (STOC'87), ACM, 1987, pp. 218–229.
[25] Louis C. Guillou, Jean-Jacques Quisquater, A "paradoxical" identity-based signature scheme resulting from zero-knowledge, in: Proceedings of Advances in Cryptology – CRYPTO'88, LNCS, vol. 403, Springer-Verlag, 1988, pp. 216–231.
[26] Gene Itkis, Leonid Reyzin, Forward-secure signatures with optimal signing and verifying, in: Proceedings of Advances in Cryptology – CRYPTO 2001, LNCS, vol. 2139, Springer-Verlag, 2001, pp. 332–354.
[27] Stanislaw Jarecki, Efficient Threshold Cryptosystems, Ph.D. thesis, MIT, 2001.
[28] Stanislaw Jarecki, Anna Lysyanskaya, Adaptively secure threshold cryptography: Introducing concurrency, removing erasures, in: Proceedings of Advances in Cryptology – EUROCRYPT 2000, LNCS, vol. 1807, Springer-Verlag, 2000, pp. 221–242.
[29] Jesper Buus Nielsen, Jesús F. Almansa, Ivan Damgärd, Simplified threshold RSA with adaptive and proactive security, in: Proceedings of Advances in Cryptology – EUROCRYPT'06, LNCS, vol. 4004, Springer-Verlag, 2006, pp. 593–611.
[30] Brian King, Improved methods to perform threshold rsa, in: Proceedings of Advances in Cryptology – ASIACRYPT 2000, LNCS, vol. 1976, Springer-Verlag, 2000, pp. 359–372.
[31] Leslie Lamport, Robert Shostak, Marshall Pease, The byzantine generals problem, ACM Trans. Program. Lang. Syst. 4 (3) (1982) 382–401.
[32] Li-Shan Liu, Cheng-Kang Chu, Wen-Guey Tzeng, A threshold GQ signature scheme, in: Proceedings of Applied Cryptography and Network Security (ACNS 2003), LNCS, vol. 2846, Springer-Verlag, 2003, pp. 137–150.
[33] Marshall Pease, Robert Shostak, Leslie Lamport, Reaching agreement in the presence of faults, J. ACM 27 (2) (1980) 228–234.
[34] Torben P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: Proceedings of Advances in Cryptology – CRYPTO'91, LNCS, vol. 576, Springer-Verlag, 1991, pp. 129–140.
[35] Tal Rabin, A simplified approach to threshold and proactive RSA, in: Proceedings of Advances in Cryptology – CRYPTO'98, LNCS, vol. 1462, Springer-Verlag, 1998, pp. 89–104.
[36] Victor Shoup, Practical threshold signatures, in: Proceedings of Advances in Cryptology – EUROCRYPT 2000, LNCS, vol. 1807, Springer-Verlag, 2000, pp. 207–220.
[37] Zhi-Jia Tzeng, Wen-Guey Tzeng, Robust forward signature schemes with proactive security, in: Proceedings of the Public-Key Cryptography (PKC'01), Springer-Verlag, 2001, pp. 264–276.
[38] Lloyd R. Welch, Elwyn R. Berlekamp, Error correction of algebraic block codes, US Patent No. 4,633,470, December 1986.
[39] Andrew Chi-Chih Yao, Protocols for secure computations, in: Proceedings of 23th Annual Symposium on Foundations of Computer Science (FOCS'82), IEEE Press, 1982, pp. 160–164.